CLAIMS

1.   (Previously Presented)   A method comprising:

collecting initial entropy data, wherein the initial entropy data includes central processing unit data and operating system data, wherein the central processing unit data comprises:

> (i) a timestamp counter;

> (ii) a number of cache misses per second;

> (iii) a number of branch mispredictions per second;

> (iv) power fluctuations;

> (v) a clock speed at which a central processing unit (CPU) is running; or

> (vi) CPU-specific counters;

storing the initial entropy data in a nonvolatile memory;

updating the initial entropy data stored in the nonvolatile memory with newly collected entropy data; and

generating a string of random bits from the updated entropy data stored in the nonvolatile memory, wherein generating a string of random bits includes:

> (i) producing a first result by hashing the updated entropy data with a first hashing algorithm;

> (ii) producing a second result by hashing the updated entropy data with a second hashing algorithm that is different from the first hashing algorithm; and

> (iii) concatenating the first result with the second result.

2.   (Canceled)

3.   (Canceled)

4.   (Previously Presented)   A method as recited in claim 1 wherein the initial entropy data includes operating system state information.

5.   (Canceled)

6.   (Previously Presented)   A method as recited in claim 1 wherein the initial entropy data is maintained in a protected portion of an operating system kernel.

7.   (Previously Presented)   A method as recited in claim 1 wherein the method is executing on a system and the initial entropy data is inaccessible by an application program executing on the system.

8.   (Canceled)

9.   (Previously Presented)   A method as recited in claim 1 wherein updating the initial entropy data stored in the nonvolatile memory includes collecting new entropy data at periodic intervals.

10.   (Original)   A method as recited in claim 1 further including communicating the string of random bits to an application program requesting a random number.

11.   (Previously Presented)   One or more computer-readable memories containing a computer program that is executable by one or more processors, the computer program causing the one or more processors to:

collect initial entropy data, wherein the initial entropy data includes central processor unit data and operating system data;

store the initial entropy data in a nonvolatile memory;

update the initial entropy data stored in the nonvolatile memory with newly collected entropy data; and

generate a string of random bits from the updated entropy data stored in the nonvolatile memory, wherein generating a string of random bits includes:

(i) producing a first result by hashing the updated entropy data with a first hashing algorithm;

(ii) producing a second result by hashing the updated entropy data with a second hashing algorithm that is different from the first hashing algorithm; and

(iii) concatenating the first result with the second result.

12.   (Previously Presented)   One or more computer-readable memories containing a computer program that is executable by one or more processors, the computer program causing the one or more processors to:

receive a request for a random number;

retrieve, from a protected portion of an operating system kernel, initial entropy data that is regularly updated with newly collected entropy data, wherein the initial entropy data includes central processing unit data and operating system data;

generate a string of random bits, wherein generating a string of random bits includes:

(i) producing a first result by hashing the updated entropy data with a first hashing algorithm;

(ii) producing a second result by hashing the updated entropy data with a second hashing algorithm that is different from the first hashing algorithm; and

(iii) concatenating the first result with the second result; and

communicate the string of random bits to the requester of the random number.

13.   (Canceled)

14.   (Previously Presented)   A method as recited in claim 12 wherein the central processing unit data includes data related to a state of a processor in a computer system and operating system data includes the state of an operating system executing on the computer system.

15.   (Canceled)

16.   (Canceled)

17. (Previously Presented) A method as recited in claim 12 wherein the updated entropy data is inaccessible by the requester of the random number.

18.-24. (Canceled)

25. (Previously Presented) An apparatus comprising:

a nonvolatile memory configured to store initial entropy data, wherein the initial entropy data stored in the nonvolatile memory is updated regularly with newly collected entropy data ; and

a random number generator, coupled to the nonvolatile memory, wherein the random number generator utilizes the updated entropy data stored in the nonvolatile memory to generate strings of random bits, wherein generating a string of random bits includes:

(i) producing a first result by hashing the updated entropy data with a first hashing algorithm;

(ii) producing a second result by hashing the updated entropy data with a second hashing algorithm that is different from the first hashing algorithm; and

(iii) concatenating the first result with the second result.

26. (Canceled)

27.   (Previously Presented)   An apparatus as recited in claim 25 wherein the initial entropy data is updated at periodic intervals.

28.   (Previously Presented)   An apparatus as recited in claim 25 wherein the updated entropy data is maintained in a protected portion of an operating system kernel such that the entropy data is inaccessible by an application program.

29.   (Canceled)

30.   (Canceled)

31.   (Previously Presented)   An apparatus as recited in claim 25 further including a timer coupled to the random number generator, the timer indicating when to update the updated entropy data stored in the nonvolatile memory device.

32.   (Currently Amended)   One or more ~~computer-readable~~ computer storage media having stored thereon a computer program that, when executed by one or more processors, causes the one or more processors to:

collect entropy data from the one or more processors and one or more operating systems executed by the one or more processors;

store the collected entropy data in a nonvolatile memory;

update the entropy data stored in the nonvolatile memory with newly collected entropy data; and

produce a string of random bits from the entropy data stored in the nonvolatile memory, wherein producing a string of random bits includes:

(i) producing a first result by hashing the updated entropy data with a first hashing algorithm;

(ii) producing a second result by hashing the updated entropy data with a second hashing algorithm that is different from the first hashing algorithm; and

(iii) concatenating the first result with the second result; and

wherein the entropy data from the one or more processors comprises:

(i) a timestamp counter;

(ii) a number of cache misses per second;

(iii) a number of branch mispredictions per second;

(iv) power fluctuations;

(v) a clock speed at which a processor is running; or

(vi) one or more processors-specific counters.

33.   (Canceled)

34.   (Currently Amended)   One or more ~~computer-readable~~ <u>computer storage</u> media as recited in claim 32 wherein the entropy data is maintained in a protected portion of an operating system kernel.

35.   (Canceled)

36.   (Canceled)

37.   (Previously Presented)   One or more ~~computer-readable~~ <u>computer storage</u> media as recited in claim 32 wherein the entropy data stored in the nonvolatile memory is updated with newly collected entropy data at periodic intervals .

38.   (Canceled)